

(19) World Intellectual Property Organization
International Bureau(43) International Publication Date
25 October 2001 (25.10.2001)

PCT

(10) International Publication Number
WO 01/80558 A2

(51) International Patent Classification: H04N 7/00

(74) Agents: CHAU, Frank et al.; F. Chau & Associates, LLP,
Suite 501, 1900 Hempstead Turnpike, East Meadow, NY
11554 (US).

(21) International Application Number: PCT/US01/40452

(22) International Filing Date: 7 April 2001 (07.04.2001)

(25) Filing Language: English

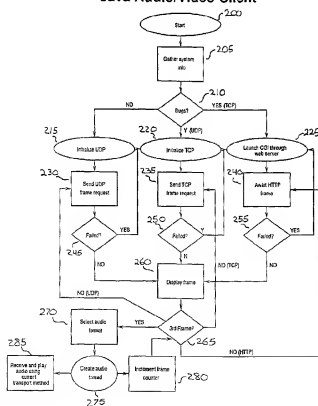
(26) Publication Language: English

(30) Priority Data:
09/548,989 14 April 2000 (14.04.2000) US(71) Applicant: SOLIDSTREAMING, INC. [US/US]; 32nd
floor, 80 Pine Street, New York, NY 10005 (US).(72) Inventor: BASTONE, Daniel; Apt. 4A, 123 East 54th
Street, New York, NY 10022 (US).(81) Designated States (national): AE, AL, AM, AT, AU, AZ,
BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE,
ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP,
KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD,
MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD,
SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN,
YU, ZA, ZW.(84) Designated States (regional): ARIPO patent (GH, GM,
KE, LS, MW, MZ, SD, SI, SZ, TZ, UG, ZW), Eurasian
patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European
patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE,
IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF,
CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: A SYSTEM AND METHOD FOR MULTIMEDIA STREAMING

Java Audio/Video Client



(57) Abstract: A system is provided for delivering streaming multimedia from a server to users via a communication network. Embedded clients at the users make requests for single datagrams. The server adapts to the variable bandwidths of the users and sends individual datagrams in response to the requesting user at the rate of available bandwidth of the requesting user.

WO 01/80558 A2



Published:

— without international search report and to be republished upon receipt of that report

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

A SYSTEM AND METHOD FOR MULTIMEDIA STREAMING

BACKGROUND OF THE INVENTION1. Field of the Invention

5 The present invention relates to a device and method for delivering video and/or audio data. More particularly to a device and method for delivering video and audio data in real time (streaming) through a communication network.

10 2. Description of Related Art

 As high bandwidth medium such as DSL, cable, and T1 lines becomes more readily available for users of communication networks such as the Internet, content providers quickly move to take advantage of the increased
15 bandwidth to deliver livelier and snazzier content. Countless websites are capable of delivering live (streaming) video or multimedia in real time. Internet users having a high bandwidth connection medium and a desktop PC with a higher speed processor receive the
20 streaming video with little difficulty. Typically, software receivers such as the Real Player by Real Networks or Media Player by Microsoft are installed on the users' PCs. The receivers receive and decode the encoded video data

delivered by content provider websites and display the decoded data as streaming video to the user.

At the content providers end, the video is first compressed or encoded. A popular video/audio compression format is MPEG, which is the format decoded by Microsoft's Media Player. MPEG (Moving Picture Experts Group) format is used in, for example, the Real Player and the Microsoft Media Player.

The MPEG technique for compressing digital video includes use of Discrete Cosine Transform (DCT), Quantization, Huffman coding, and Motion Compensated Predictive coding, in which the differences in what has changed between an image and its proceeding image are calculated and only the differences are encoded. Predictive coding requires interframe processing, i.e., data from neighboring frames are needed to successfully encode and decode an image; therefore, individual frames must be temporarily stored in a buffer and the image is encoded and decode using multiple frames. Buffering allows a server to send data at a constant rate, regardless of the rate at which the client displays the data. A disadvantage of MPEG and the buffering technique is that a considerable amount of memory is needed for buffering. In portable devices, sufficient memory may not exist.

Another video compression technique, known as JPEG (Joint Photographic Expert Group), employs the MPEG coding process except predictive coding. Thus, JPEG compression does not rely on interframe processing, i.e., each frame is
5 independently processed and has no processing relationship to another frame. Therefore, JPEG compression does not require frame buffering.

Various methods and protocols for delivering data are available depending on the bandwidth of the connecting
10 medium. One example is the Transmission Control Protocol (TCP). The TCP typically functions in conjunction with the Internet Protocol (IP). The TCP provides reliable, stream-oriented connections that hide most of IP's shortcomings; i.e., the basic nature of IP cannot guarantee
15 the data will be delivered correctly. The TCP/IP protocol suite gets its name because the TCP protocol is layered on top of the IP protocol. The TCP layer interfaces on one side to application processes and on the other side to the IP protocol.

TCP data is organized as a stream of bytes, much like a
20 file. The datagram nature of the network is concealed. A mechanism (the Urgent Pointer) exists to let out-of-band data be specially flagged. Sequence numbers are used to coordinate which data has been transmitted and received.
25 TCP will arrange for retransmission if it determines that

data has been lost. This method provides for reliable delivery. TCP will dynamically learn the delay characteristics of a network and adjust its operation to maximize the throughput without overloading the network, this gives TCP the quality of network adaptation. TCP manages data buffers, and coordinates traffic so its buffers will not overflow. Fast senders will be stopped periodically to keep up with slower receivers, resulting in flow control.

TCP operates in both directions (full duplex) and in an almost completely independent manner, akin to two independent byte streams traveling in opposite directions. No TCP mechanism exists to associate data in the forward and reverse byte streams. Only during connection start and close sequences can TCP exhibit asymmetric behavior, i.e., data transfer in the forward direction but not in the reverse, or vice versa.

Each endpoint of a TCP connection will have a buffer for storing data that is transmitted over the network before the application is ready to read the data. This lets network transfers take place while applications are busy with other processing, improving overall performance. To avoid overflowing the buffer, TCP sets a Window Size field in each packet it transmits. This field contains the amount of data that may be transmitted into the buffer. If this number falls to zero, the remote TCP can send no more data. It must

wait until buffer space becomes available and it receives a packet announcing a non-zero window size.

Sometimes, the buffer space is too small. This happens when the network's bandwidth-delay product exceeds the buffer size. The simplest solution is to increase the buffer, but for extreme cases the protocol itself becomes the bottleneck (because it doesn't support a large enough Window Size). Under these conditions, the network is termed an LFN (Long Fat Network).

When a host transmits a TCP packet to its peer, it must wait a period of time for an acknowledgment. If the reply does not come within the expected period, the packet is assumed to have been lost and the data is re-transmitted. The time that the protocol will wait for a reply is a variable. Over an Ethernet, no more than a few microseconds should be needed for a reply. If the traffic must flow over the wide-area Internet, a second or two might be reasonable during peak utilization times. If a communication device is on a satellite traveling toward Mars, minutes may be required before a reply.

Round-Trip Time (RTT) estimates are an important performance parameters in a TCP exchange, especially when dealing with an indefinitely large transfer. All TCP implementations eventually drop packets and retransmit them, no matter the quality of the link. If the RTT estimate is

too low, packets are re-transmitted unnecessarily; if too high, the connection can sit idle while the host waits to timeout.

The User Datagram Protocol (UDP) is used in higher bandwidth communication links. UDP is a connectionless protocol that, like TCP, runs on top of an IP network. Unlike TCP/IP, UDP/IP provides very few error recovery services, offering instead a direct way to send and receive datagrams over an IP network. UDP is used primarily for broadcasting messages over a network. UDP packets are delivered like IP packets; connectionless datagrams that may be discarded before reaching their targets. UDP is useful when TCP would be too complex, too slow, or just unnecessary.

UDP provides a few functions beyond that of IP. For example, Port Numbers. UDP provides 16-bit port numbers to let multiple processes use UDP services on the same host. A UDP address is the combination of a 32-bit IP address and the 16-bit port number. Unlike IP, UDP does checksum its data, ensuring data integrity. A packet failing checksum is simply discarded, with no further action taken.

A common gateway interface (CGI) is one way for a web server to pass a web user's request to an application program, which in turn passes data back to be forwarded to the user. When the user requests a web page (for example, by

clicking on a hypertext link), the server retrieves the requested page, sending the page to the client. However, when a user submits a form on a web page, it usually needs to be processed by an application program. The web server typically passes the form information to a small application program (applet) that processes the data and may send back a confirmation message. This method for passing data between the server and the application is called the common gateway interface (CGI). The CGI is part of the web's Hypertext Transfer Protocol (HTTP).

For a client system having a lesser connection bandwidth, such as connection through a 56K modem, specific communication protocols can be established by an application program predownloaded or installed in the client's computer. A CGI can be used to pass a client's request to the application program. The CGI provides a consistent way for data to be passed from the user's request to the application program and back to the user. In other words, CGI operates in conjunction with clients and servers regardless of which operating system (OS) is being used by the parties, for example, Windows, Macintosh, UNIX and Linux, OS/390, or others. A CGI application may be written in a number of different languages.

An alternative to a CGI application is Microsoft's Active Server Page (ASP), in which a script embedded in a web page is executed at the server before the page is sent.

In a desktop client environment in which memory speed
5 and connection bandwidth are sufficient, the connection rate is stable and servers can 'push' the MPEG-type datagrams to the clients synchronously, i.e., frames are buffered at the server and the buffer content is dumped or transmitted to the clients at a substantially periodic rate. At the
10 desktop PC, the datagrams are also buffered because a single image depends on several datagrams.

As wireless applications and devices grow in popularity, the limitations of wireless applications and devices such as narrow bandwidth and limited memory and
15 processing capacity must be addressed. In particular, content providers who wish to deliver substantially the same streaming video contents to wireless users as desktop users must find a viable solution to overcome these limitations. Therefore, a need exists for a system and method for
20 delivery of streaming media to the client regardless of the client's available bandwidth.

SUMMARY OF THE INVENTION

These and other objects, features, and advantages of
25 the present invention will become apparent from the

following detailed description of illustrative embodiments thereof, which is to be used in connection with the accompanying drawings.

A method for streaming video data over a network in real time is provided. The method includes initializing a transport mode for the video data, sending a data request for a single frame of video data from a client to a server, retrieving the single frame from a memory at the server, and sending the video data to the client.

The step of initializing also includes listing available transport modes for the client, determine whether incompatibilities exist between the available transport modes and software, choosing a transport mode from the list, and initializing parameters of the transport mode at the client for client control of video steaming.

Initializing is performed by a client application which is capable of running in different operating systems. Alternatively, initializing is performed by a client embedded in a web page. The client embedded in the web page can be a common gateway interface and an active server page. The transport mode is chosen from the following, a UDP, a TCP, and an HTTP, however other modes are contemplated.

The steps of sending a data request for a single frame from the client to a server, retrieving the video data from

a shared memory, and sending the retrieved video data to the client, are repeated for each video frame.

Storing video includes capturing a thread from a specified source, and storing the captured thread in the server's shared area of memory.

According to another aspect of the present invention, a storage medium having a stored program which is executable by a processor for causing the processor to perform method steps for streaming video communication is also provided.

The method steps comprising requesting a packet representing a single datagram from a server over a communication network, receiving a requested packet, processing and displaying the requested packet, incrementing a datagram frame counter, requesting a next packet based on the frame counter value from the server, and asynchronously processing and displaying the next packet when received.

The communications between the processor and the server is preferably by Wireless Application Protocol (WAP). The server is accessed by said processor via HTTP. The packet representing a datagram is preferably JPEG encoded, and the step of asynchronously processing and displaying is independent of data from the step of processing and displaying the requested packet.

An apparatus is also provided for communicating streaming video data between a plurality of users and a

server connected by a communication network, comprising a stored program executable by a processor in said server for causing the server to receive requests for individual datagrams from the plurality of users and forwarding individual datagrams in response to each request to the user making the request at a rate based on available bandwidth of the user making the request.

BRIEF DESCRIPTION OF THE DRAWINGS

The preferred embodiments are described with reference to the drawings wherein:

FIG. 1 is a diagram of a system for streaming data according to one embodiment of the invention;

FIG. 2 is a flow diagram of a preferred embodiment of the invention for streaming audio/video data to a client;

FIG. 3 is a flow diagram of a preferred embodiment of the invention for streaming video data;

FIG. 4 is a flow diagram of a preferred embodiment of the invention for streaming audio data; and

FIG. 5 is a flow diagram of a method for streaming audio data over a CGI HTTP transport mode.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Preferred embodiments of the present invention will be described below in more detail with reference to the accompanying drawings.

5 The present invention relates to a system and method for streaming video. The invention is implemented over a network of processors. The network can include, a local area network (LAN), a wide area network (WAN), an Intranet, an Internet, a wireless network, or the like. These networks
10 can be in any configuration including, for example, star, ring, and bus.

For purposes of the present invention a client will, by definition, be receiving and displaying data, while the server will be providing data to the client. A system and
15 method of the present invention enables the client to receive audio/video data in real time regardless of the bandwidth available to the client. This is achieved through the implementation of a dynamic bandwidth adaption method for managing the flow of data. As a result, devices such as,
20 PDAs, hand-held PCs, and various mobile devices with little bandwidth are able to receive streaming video in real time. It is readily appreciated by one ordinarily skilled in the art that the invention is not limited to these devices and is also suitable to desktop computers, servers, and the
25 like.

A system and method according to the present invention can be construed as a browser that allows a client to access the server's network to receive streamed data content. The browser is preferably WAP (Wireless Application Protocol) compatible, but can be deployed to any device including those which are not WAP compatible. The browser preferably adapts to different codecs which do not require interframe processing, for example, Motion JPEG and Wavelet, allowing the flexibility to port software to embedded devices having limited memory resources. Frame buffering at the server can be dispensed with because each frame can be independently processed and delivered.

As shown in FIG. 1, the server 110 having audio/video data is connected to a network 100, in this example a bus network. While FIG. 1 depicts a bus network 100, any other network capable of supporting a server and client is contemplated by the invention. Alternatively, both the server and client may be embodied in the same computer and operate without a network. Typically a server is a computer program that provides a service to another computer program, the client. A server can function as a client and a client as a server depending on whether services are being offered or requested. The server, for purposes of the invention, is a stored program including a script for downloading audio/video data and an applet. A client 120 or

130 is also connected to the network. The client 120 captures the data from the server 110. The capture can take place from a local capture card, a local looped file, a local file on-demand, or a remote IP address. Each will be explained below in more detail. It should be readily apparent to one ordinarily skilled in the art that the client is described for client 120 but the embodiment is applicable for multiple clients.

The client 120 is a script which may be stored in the memory of a Personal Digital Assistant (PDA), PC or embedded in a web page. The client is a application which is capable of running in different operating systems, for example, Windows CE, Palm OS, Nokia PDA's, Windows, Linux etc. Alternatively, the client is embedded in a web page, for example, a common gateway interface (CGI) or a Microsoft Active Server Page (ASP). A CGI may be written in different programming languages, for example, C, C++, Java, and Perl, though this is not a complete list of possible applicable languages. Execution of the client ensures cross-platform and cross-browser functionality. Referring to FIG. 2, upon startup 200, the client gathers information 205 regarding its own operating environment needed to choose available transport modes. The information is checked against known bugs or incompatibilities 210 with certain operating systems (OS) and browser combinations. An example of a bug includes,

the Java Interpreter used in Microsoft Internet Explorer v4.0 which has broken implementation of User Data Protocol (UDP). An appropriate transport mode is chosen according to the incompatibilities 210. The client will detect this and
5 default to suitable protocol, for example, Transmission Control Protocol (TCP) or Hypertext Transfer Protocol (HTTP) modes. The method of connection, e.g.UDP, TCP, HTTP, are referred to as a transports. The client initializes parameters which control the transport mode. This allows the
10 same client to operate all possible configurations, for example, audio, video, audio/video, on-demand, live, or other similar configurations.

Assuming that no incompatibilities have been detected, the client attempts a UDP connection 215 to the server to
15 retrieve a frame of video 230. In one embodiment, if the connection fails 245, a second attempt 220 is made to establish a UDP connection. After a specified number of attempts to establish a UDP connection, a TCP connection will be attempted. Note that in the present example, UDP is
20 preferable to TCP connections based on bandwidth, however other connections may provide superior bandwidth than UDP, in such a case the wider bandwidth connection would be attempted first and the UDP second. In an alternative embodiment, an attempt to establish a TCP connection is made
25 after the first failed UDP connection attempt.

Likewise, if the TCP connection fails 250 to retrieve a frame 235 then an attempt to establish a HTTP connection 225 is made. Alternatively, more than one attempt to establish a TCP connection will be made. Further attempts to establish a connection are made with progressively narrower transports until the transport mode with a narrowest acceptable bandwidth has been reached. For example, a HTTP connection. Attempts to establish this transport mode will be continue until a connection is established, 240/255. At this point it is assumed that the server is down or the network has failed. When the server comes up or the network improves, video will begin streaming once the connection has been established. In an alternative embodiment, the transport mode with the narrowest acceptable bandwidth will be attempted a specified number of times.

Once a successful video connection has been established and an audio format has been selected, an audio connection will be made and display the data will begin 260.

If at any point a connection fails 265, a connection will be attempted with the next bandwidth, for example from UDP to TCP. While in the alternative transport mode, the mode with a lower bandwidth, periodic attempts to establish a connection with the original transport will be made. No user intervention is needed during operation of the method.

Alternatively, if the connection is maintained, the datagram frame counter is incremented 280. A request is then made for the next packet based on the frame counter value from the server. The steaming video data will continue until the client terminates the connection with the server.

If displaying a stream on-demand, the client is given the ability to fast forward, rewind, and jump to arbitrary parts of the stream. A clock is also displayed indicating the current location in the stream. The client initially sends a query to the server requesting the length of the stream in frames. Proceeding with each frame, the server sends the frame number, allowing the client's clock to remain accurate regardless of the speed of the stream. When the user wishes to fast forward, rewind, or jump to a specific location in the stream, it sends a request with the desired target frame number to the server. The server responds by seeking to the requested frame number and streaming from there. All other functions of rate adaption and protocol selection operate as normal.

An advantage to the present invention, related to the client's ability to move on the stream, is error resilience. That is, since each video frame is independent, if an error occurs in a frame the invention can either attempt to fix the error or drop the frame. When a frame is dropped, the invention requests the next frame and preserves the

continuity of the video and/or audio display. This is especially advantageous in devices having limited processing power, and therefore, lacking the resources to fix errors.

Having described the role of the client above, the
5 method will be described in reference to various types of capture. Other types of capture may also be implemented in the present invention.

The server may capture a thread 305 from a specified source, e.g., a local capture card. Example of a local
10 capture card include the SunVideoPlus Osprey and the SunVideo (sun) capture board. In this instance, the server makes use of native Solaris threads to achieve rate-adaptive connections to the clients. A thread is a placeholder information associated with a single use of a program that
15 can handle multiple concurrent users. From the program's point-of-view, a thread is the information needed to serve one individual user or a particular service request. If multiple users are using the program or concurrent requests from other programs occur, a thread is created and
20 maintained for each of them. The thread allows a program to know which user is being served as the program alternately gets re-entered on behalf of different users. One way thread information is kept is by storing it in a special data area and putting the address of that data area in a register. The
25 OS always saves the contents of the register when the

program is interrupted and restores it when it gives the program control again.

On startup, the server creates two threads which capture video 300 and audio 400 from the specified source. The server places the captured data 310 into a shared area of memory 315 which is accessible to all other threads within the server. Other threads are created which accept incoming requests for each transport type. These incoming requests for each client are handled as part of a non-blocking loop. Each transport type is handled differently.

For example, for UDP video 325 connections, the non-blocking loop accepts and services connections. Since UDP is "connectionless," there is no need to maintain a persistent connection to the client and each client connection is really a request for a single frame. The thread receives a single byte datagram from a client 340, immediately retrieving 355 and sending 370 back the frame currently stored in the shared memory segment 315. A datagram is a self-contained, independent entity of data carrying sufficient information to be routed from the source to the destination computer without reliance on earlier exchanges between this source and destination computer and the transporting network. This process continues indefinitely.

Since sending a UDP datagram is a non-blocking operation, the server need only have one thread. Thus, the

amount of CPU time and memory required from UDP is drastically less than that needed for other connections.

Another example is TCP connections 330. The server waits for a new connection to be established with a client 345. A non-blocking loop accepts incoming requests. The client corresponding to the requests are arranged in a connection pool list. Since TCP is a connected protocol, an independent thread is then created to service 360 each client. The service threads act like the UDP thread, awaiting 375 a single byte (request) from the client, the requested frame is retrieved 385 by the server from the shared memory 315. The retrieved frame is then sent 390 to the client.

The above two examples are rate adaptive solutions. By allowing the clients to control the flow of video frames rather than simply pushing each frame to each client as it is captured, data bottlenecks typically associated with streaming media over the Internet are eliminated.

As an illustration, two desktop PCs are connected to a server. The first by 100Mbps Ethernet, the second by a 28.8Kbps dialup Internet connection. The 100Mbps machine will receive video at the rate it is captured, up to 30fps. The rate of speed of the video is due in great part to the available bandwidth, the 100Mbps sends out frame requests to the server fast enough to allow for a 30 fps stream.

However, the 28.8Kbps machine requires more time to receive each frame and therefore sends out frame requests less frequently. Thus the rate of speed is about 3 to 4fps, and both machines will be at the same place in the stream. The illustration method of the present invention accomplishes this through client side requests for current frames. These requests may or may not be for the next sequential frame. Comparing the machines, the 100Mbps machine will show a higher quality video, while the 28.8Kbps machine will appear to be skipping frames in order to maintain a real time stream.

This approach has several advantages over conventional streaming methods. Current methods require re-encoding the video at several frame rates in order to support users with different bandwidth availability. The user is also required to select a rate beforehand that is appropriate for their connection. In contrast, the illustrative method according to the present invention needs only one rate of capture and encoding. A user no longer needs to know anything about their network bandwidth. The user will connect to the stream at a rate which adapts itself to the environment. Additionally, degraded network conditions will not create a bottleneck and will not cause the stream to fall behind real time, rather the frame rate will decrease and more frames will be skipped. When conditions improve, so will the frame

rate. Further, this approach will take advantage of high-bandwidth consumer access devices, such as xDSL and 38GHz wireless connections without any changes or updating the server of client.

5 Other transport modes are supported by the invention, for example, HTTP 335. HTTP is a one-way protocol and thus cannot perform true rate adaption. It is included for clients that do not have UDP, TCP, or another transport available to them. This may be due to a network firewall, or
10 a packet filter for example. Further, some Internet Service Providers (ISP) may not support connections on certain ports, or not support UDP at all. HTTP connections are achieved through the use of an external program which is activated by a web server 500, e.g. common gateway interface
15 (CGI). A main server creates one thread to accept and service connections from this external program via local UNIX domain sockets and UDP 505. Although UDP is used here, it is only for local redirection of frames from the main server to the CGI. The server waits for a CGI request 350.
20 The CGI requests frames at a steady rate from the main server 510. As the main server retrieves frames 365 from the shared memory 315, the frames are re-broadcast to the client 380 back through the web server 520 via HTTP. To avoid bottlenecks, from the audio push 525, this transport mode

transmits 380/515 at a default rate of 1fps in order to remain real time regardless of available bandwidth.

Another capture can be from a local file which is looped. This method obtains video and audio data from a single file that has been pre-encoded from a capture board and stored on the server. It operates exactly as above and uses the exact same thread structure. Data is read in from the specified file and placed into the shared memory segments at the rate it was encoded. When the end of the file is reached, it re-starts from the beginning.

Still another capture can be had from a local file on-demand. This method deals with more than one source of audio/video data being sent to all clients. Each client needs its own unique copy of the server which then acts as described above using the requested file. This is accomplished by creating another layer of threads which create entire server sets of threads for each client. While this creates a high amount of overhead, the data is being delivered in an compressed and encoded state, therefore it is not necessary to make deliveries in real time. In addition, the server returns the number of frames in the stream to the client as well as the frame number of each frame sent, therefore, the client's clock remains accurate. The server also receive requests from the client indicating that it wishes to jump forward or backward to a specific

frame, the server seeks to this location in the file and continues streaming.

Yet another capture is from a remote IP address, according to this method the server receives data from another server on a remote machine. The remote machine may be located anywhere on the network that is accessible from the local machine's network. The local server acts as a single client to the remote server, and pulls a stream from the remote server in the same rate-adaptive fashion as the Java client. The local server then places the data into its shared memory buffers, and re-broadcasts it as if it were being captured locally. All other functions of the local server operate as normal.

Because the local server is acting as a regular client, it does not matter how the remote server is captures data, it can be from any of the methods described above. It can even be capturing its data from another remote server, allowing for server chains to be created that, for example, re-broadcast a source feed on different networks.

Like video 300, audio data 400 may be transported by the methods described above: UDP 440, TCP 445, and HTTP 450. However, it is pushed data. As new audio data is captured 410, it is immediately sent out to all clients 430. This allows for a steady audio stream which will inherently be in sync with the video played in real time according to the

invention. The server also attempts to adapt the streamed audio to the clients available bandwidth. This is accomplished by the server making available multiple types of audio simultaneously, for example, raw uncompressed 410, Global System for Mobile communication (GSM) encoded audio 415, and G.728 encoded audio 420 (G728 is specified in ITU-T recommendation G.728, "Coding of speech at 16Kbit/s using low-delay code excited linear prediction"). Uncompressed audio 410 occupies 64k of bandwidth, G.728 420 occupies 16Kbps of bandwidth, and GSM 415 occupies 13Kbps of bandwidth. The client times the amount of time between two frames 265, for example, the second and third frames, and based on this time selects the appropriate available audio format 270. The server creates a thread 435 to accept incoming audio connections via each of the available transport modes: UDP 440, TCP 445, and HTTP 450. The server waits for a new connection 455 a, b, c to be established by a client. Upon connection, the client sends preferred audio format information to the server 460 a, b, c. The server then creates a service thread 275/465 a, b, c, for the client which delivers audio in the requested format. These service threads wait for a signal from the corresponding audio encoding thread 470a, b, c, that more audio has become available. The server retrieves audio data from a selected buffer 475a, b, c, then the selected buffer is sent to the

client 285/480a, b, c, and immediately wait for another signal 470a, b, c. HTTP connections will automatically default to the lowest bandwidth signal, since a degraded connection is assumed.

5 Advantageously, the system and method according to an illustrative embodiment of the invention functions in low bit rate environments, at about 9.6 Kbps, and suffers no degradation even during a transmission over a 14.4 Kbps network. Theoretically there is no upper bound as the
10 system utilizes an adaptive bandwidth method. That is, the system scales the data stream to the available bandwidth so that as bandwidth increases the stream rate will increase accordingly. The flow of data from the server to the client is intelligently managed so as to completely eliminate
15 network bottlenecks traditionally associated with streaming media. This is especially beneficial in wireless environments where available bandwidth may be limited or inconsistent. The end user is allowed to receive multimedia data without fore knowledge of the available bandwidth while
20 maintaining a real-time stream. This also frees the content provider from having to provide multiple media streams to accommodate differing connection speeds. In order to control the drain of the server's network capacity, the system and method provides for a cap. The cap is utilized by the server

to restrict usage of the stream to maintain the integrity of the server's network.

Having described preferred embodiments of a system and method for multimedia streaming, it is noted that
5 modifications and variations can be made by persons skilled in the art in light of the above teachings. It is therefore to be understood that changes may be made in the particular embodiments of the invention disclosed which are within the scope and spirit of the invention as outlined by the
10 appended claims. Having thus described the invention with the details and particularity required by the patent laws, what is claimed and desired protected by Letters Patent is set forth in the appended claims.

WHAT IS CLAIMED IS:

1. A method for streaming video data over a network
in real time, comprising:

initializing a transport mode for the video data;

5 sending from a client to a server a data request for a
single frame of video data;

retrieving the single frame from a memory at the
server; and

sending the video data to the client.

10 2. The method for streaming video in Claim 1, wherein
the steps of sending a data request for a single frame from
the client to a server, retrieving the video data from a
shared memory, and sending the retrieved video data to the
15 client, are repeated for each video frame.

3. The method for streaming video as in claim 2,
wherein said each video frame is processed for display
independently from processing of another video frame.

20 4. The method for streaming video in Claim 1, wherein
the step of initializing further comprises:

determining a list of available transport modes for the
client;

determine incompatibilities between the available
transport modes and software;
choosing a transport mode from the list; and
initializing parameters of the transport mode at the
5 client for client control of video streaming.

5. The method for streaming video in Claim 4, wherein
the step of initializing is performed by a client
application which is capable of running in different
10 operating systems.

6. The method for streaming video in Claim 4, wherein
the step of initializing is performed by a client embedded
in a web page.

7. The method for streaming video in Claim 6, wherein
the client embedded in the web page is chosen from one of a
common gateway interface and an active server page.

8. The method for streaming video in Claim 1, wherein
the transport mode is chosen from the group consisting of a
UDP, a TCP, and a HTTP.

9. The method for streaming video in Claim 1, wherein storing video further comprises:

capturing a thread from a specified source; and

storing the captured thread in the server's shared area

5 of memory.

10. A storage medium having a stored program which is executable by a processor for causing the processor to perform method steps for streaming video communication, the method steps comprising:

requesting a packet representing a single datagram from a server over a communication network;

receiving a requested packet;

processing and displaying said requested packet;

15 incrementing a datagram frame counter;

requesting a next packet based on the frame counter value from the server; and

asynchronously processing and displaying said next packet when received.

20 11. The method of claim 10, wherein communications between said processor and said server is by Wireless Application Protocol (WAP).

12. The method of claim 10, wherein said server is accessed by said processor via HTTP.

13. The method of claim 10, wherein said packet
5 representing a datagram is JPEG encoded.

14. The method of claim 10, wherein said step of asynchronously processing and displaying is independent of data from said step of processing and displaying said
10 requested packet.

15. An apparatus for communicating streaming video data between a plurality of users and a server connected by a communication network, comprising:

15 a stored program executable by a processor in said server for causing the server to: receive requests for individual datagrams from the plurality of users; and
forwarding individual datagrams in response to each request to the user making the request at a rate based on
20 available bandwidth of the user making the request.

16. The apparatus according to claim 15, wherein said plurality of users are wireless mobile devices.

17. The apparatus according to claim 15, wherein said server is accessible via HTTP.

18. The apparatus according to claim 15, wherein the
5 datagrams are JPEG encoded.

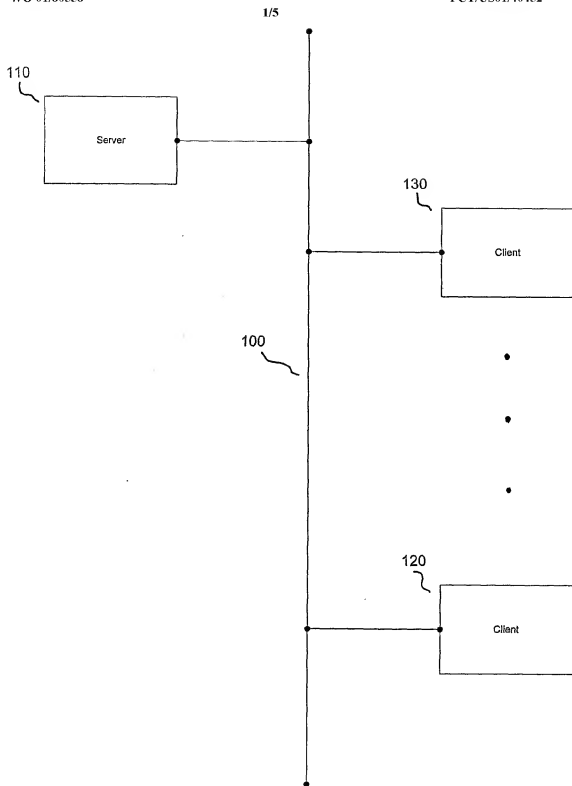


FIG. 1

Java Audio/Video Client

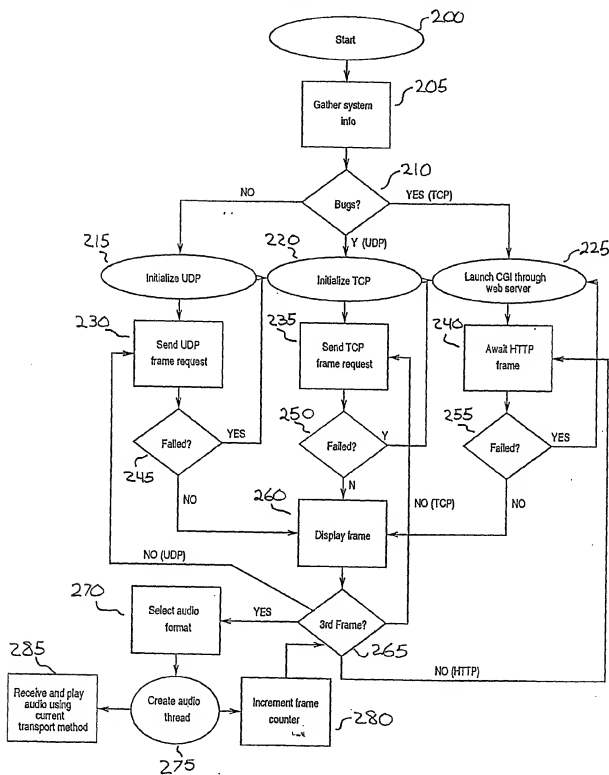


FIG. 2

Multithreaded Audio/Video Server

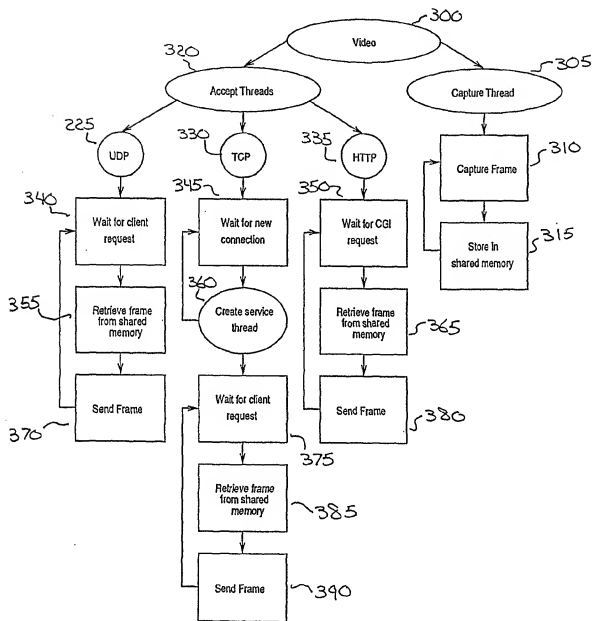


FIG. 3

Multithreaded Audio/Video Server

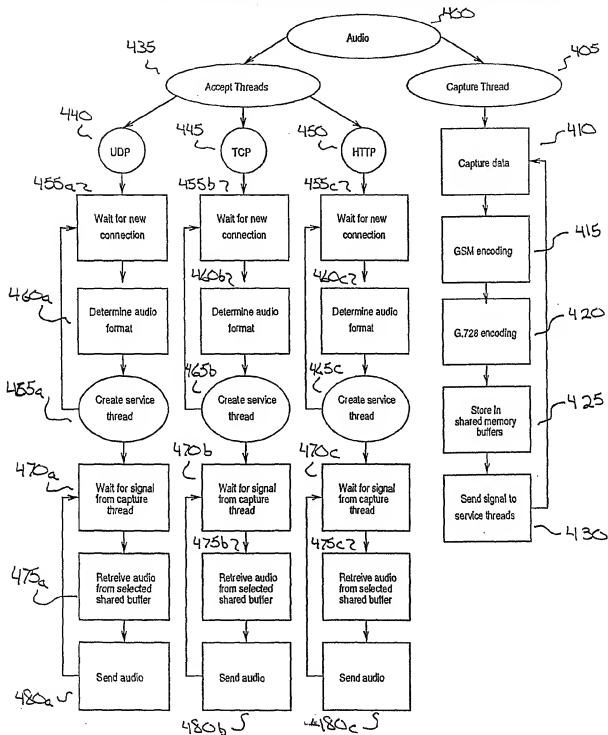


FIG. 4

CGI HTTP Transport Module

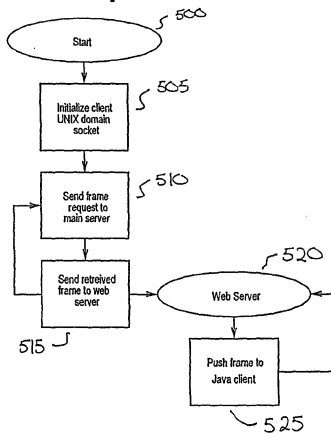


FIG. 5